(12) **UK Patent Application** (19) **GB** (11) **2 306 043** (13) **A**

(43) Date of A Publication 23.04.1997

(54) Audio synthesizer

(57) An audio synthesizer is disclosed for generating an analogue or digital audio output in response to coded control instructions representing musical events, such as a MIDI data stream (output of 310). The synthesizer comprises: a general purpose computer portion having a CPU programmed to receive the control instructions and generate audio samples; a special purpose hardware portion 330, eg a standard sound card, for receiving the control instructions and generating the audio samples; a controller 335 for directing the control instructions either so that the general purpose computer portion 340, 350 generates the audio samples or so that the hardware portion 330 generates the audio samples; and means to combine the audio samples generated by the general purpose computer portion and the hardware portion to form an audio output which accords with the control instructions.

FIG.3

GB 2 306 043 A

130

140

150

170

CPU

RAM

160

120

160

110

100

FIG. 1

UHU-35-043

FIG. 2

Uno-85-043

FIG.3

```
          ┌─────────────────────────────┐
  ┌──────►│   DETERMINE NOTE TO BE      │──── 400
  │       │  PRODUCED FOR EACH VOICE    │
  │       └─────────────────────────────┘
  │                     │
  │                     ▼
  │       ┌─────────────────────────────┐
  └───────│  PRODUCE SAMPLES            │──── 410
          │  FROM WAVE TABLES           │
          └─────────────────────────────┘
```
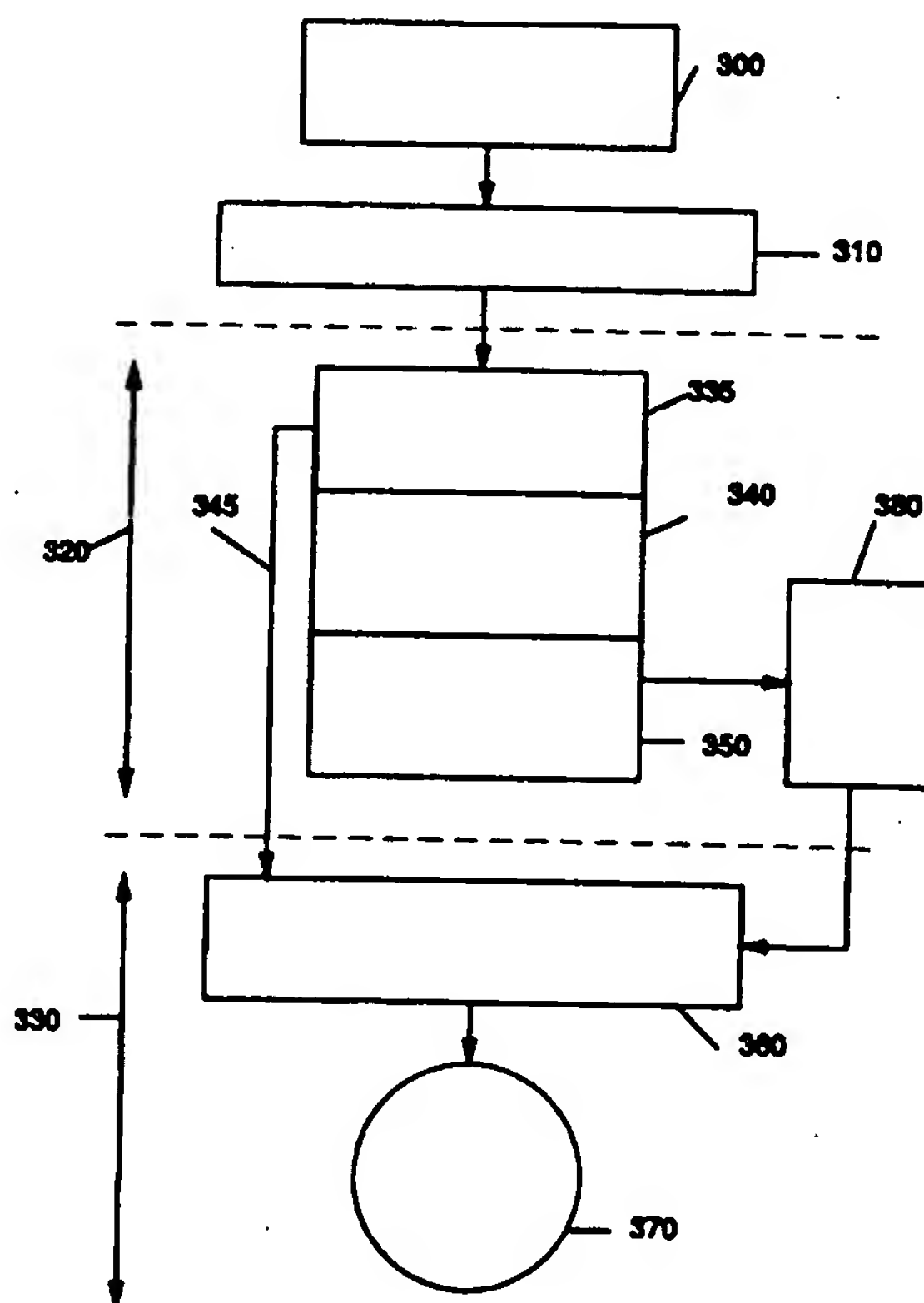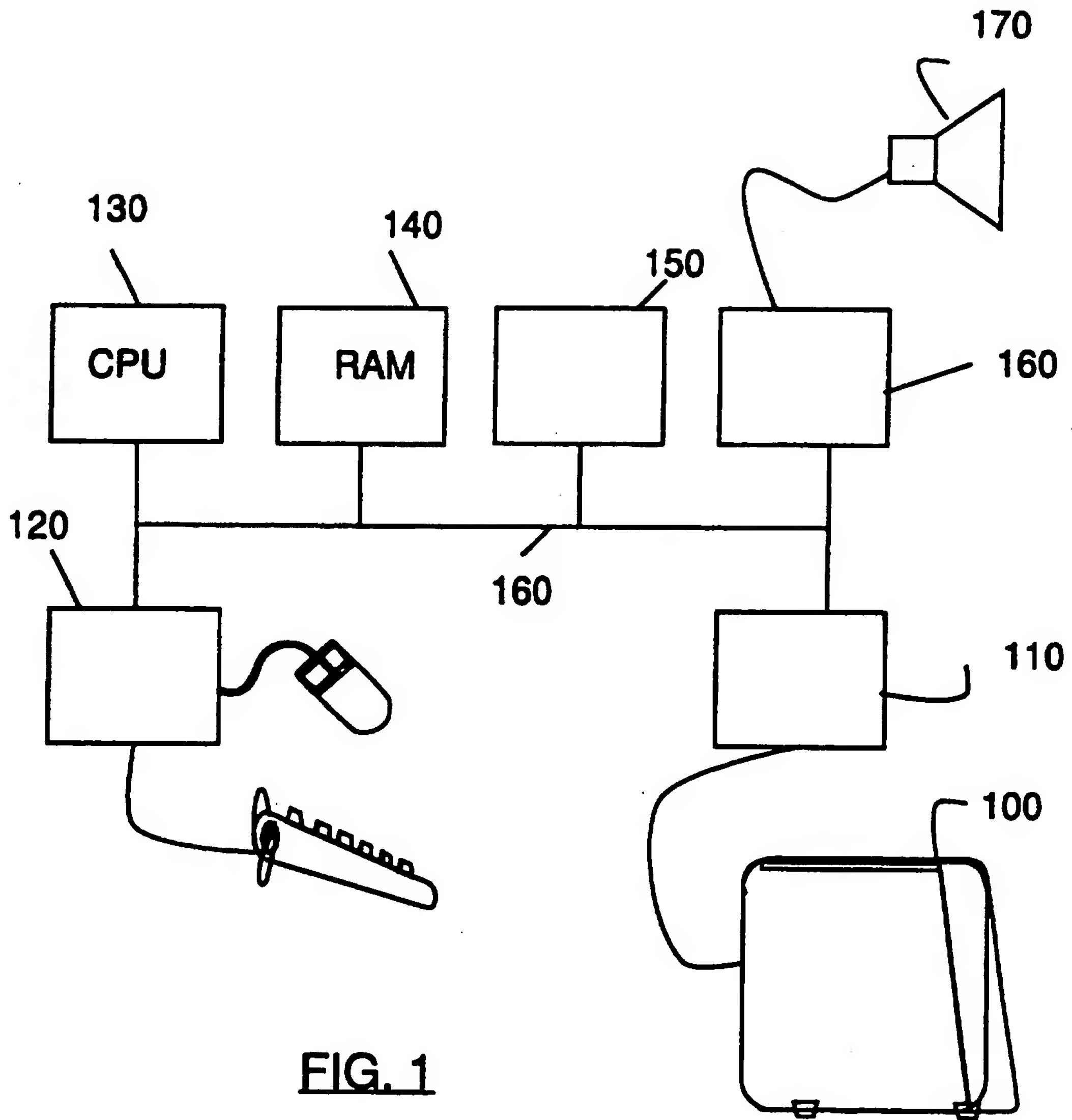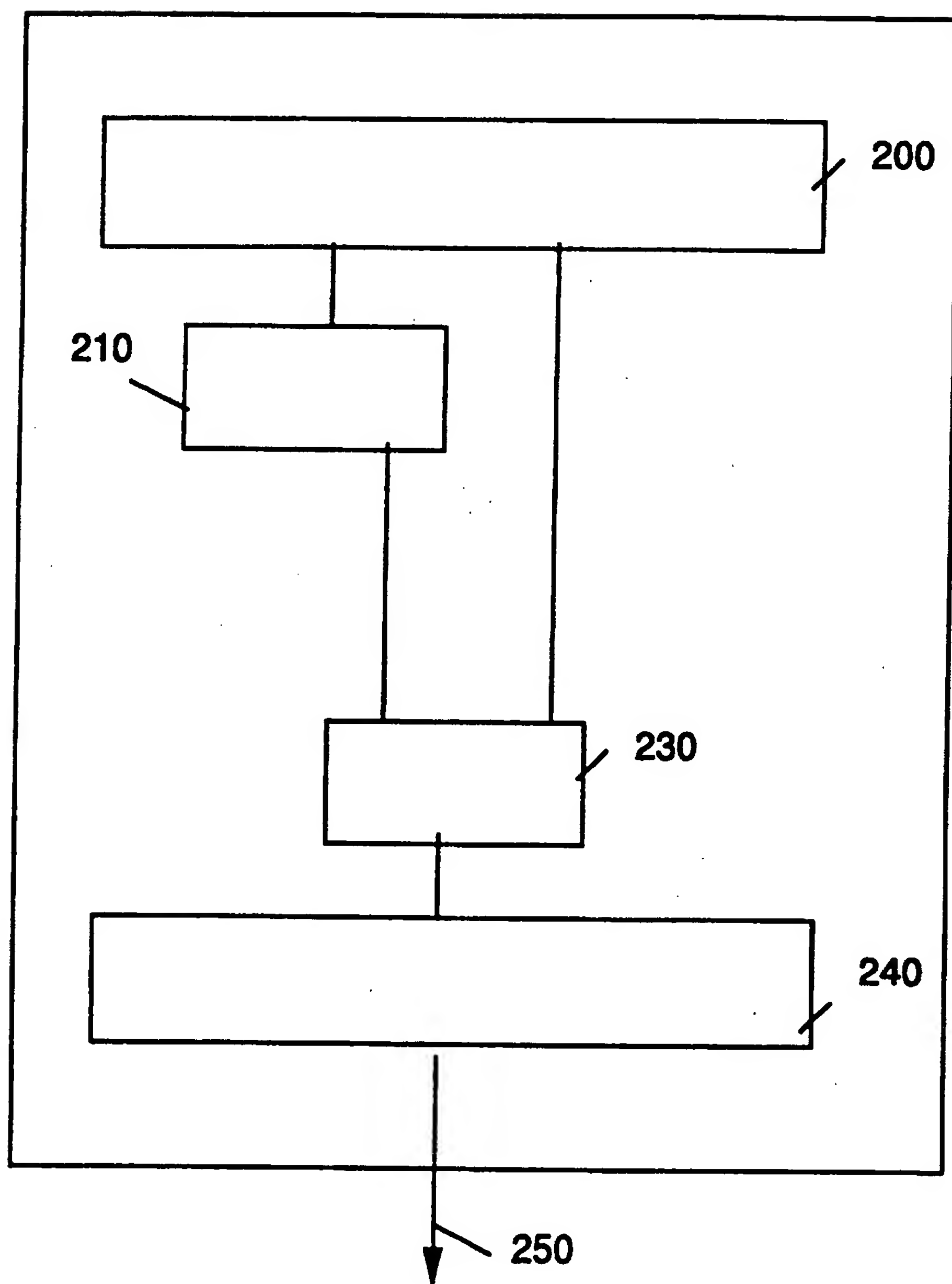
FIG.4

FIG.5

FIG. 6

AUDIO SYNTHESIZER

TECHNICAL FIELD

5   The invention relates to audio synthesizers for generating digitally encoded audio samples in response to coded control instructions representing musical events, such as a MIDI data stream.

BACKGROUND ART

10

Currently, most computers use sound cards or other dedicated hardware in order to produce music, for instance for games, composition and other multimedia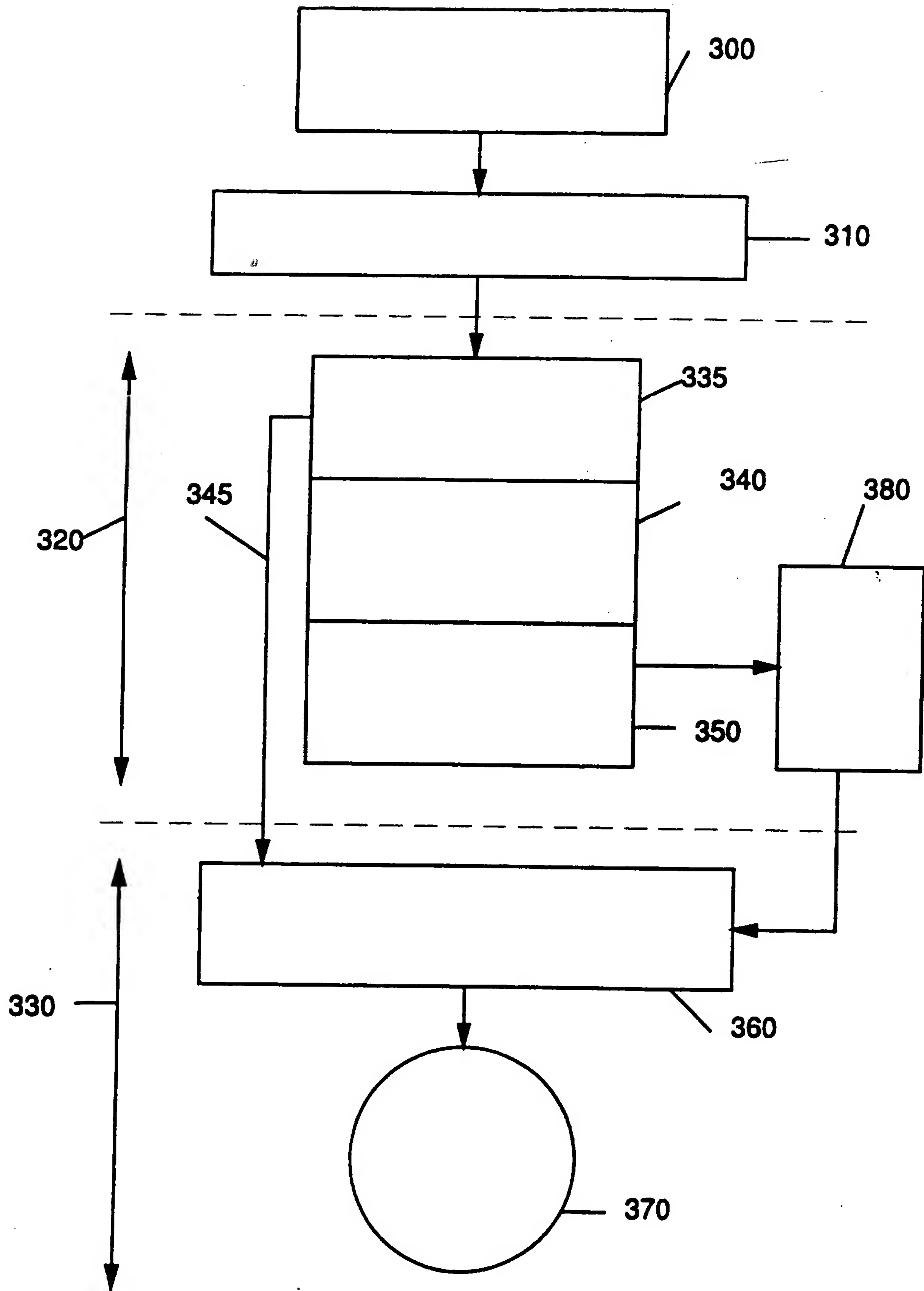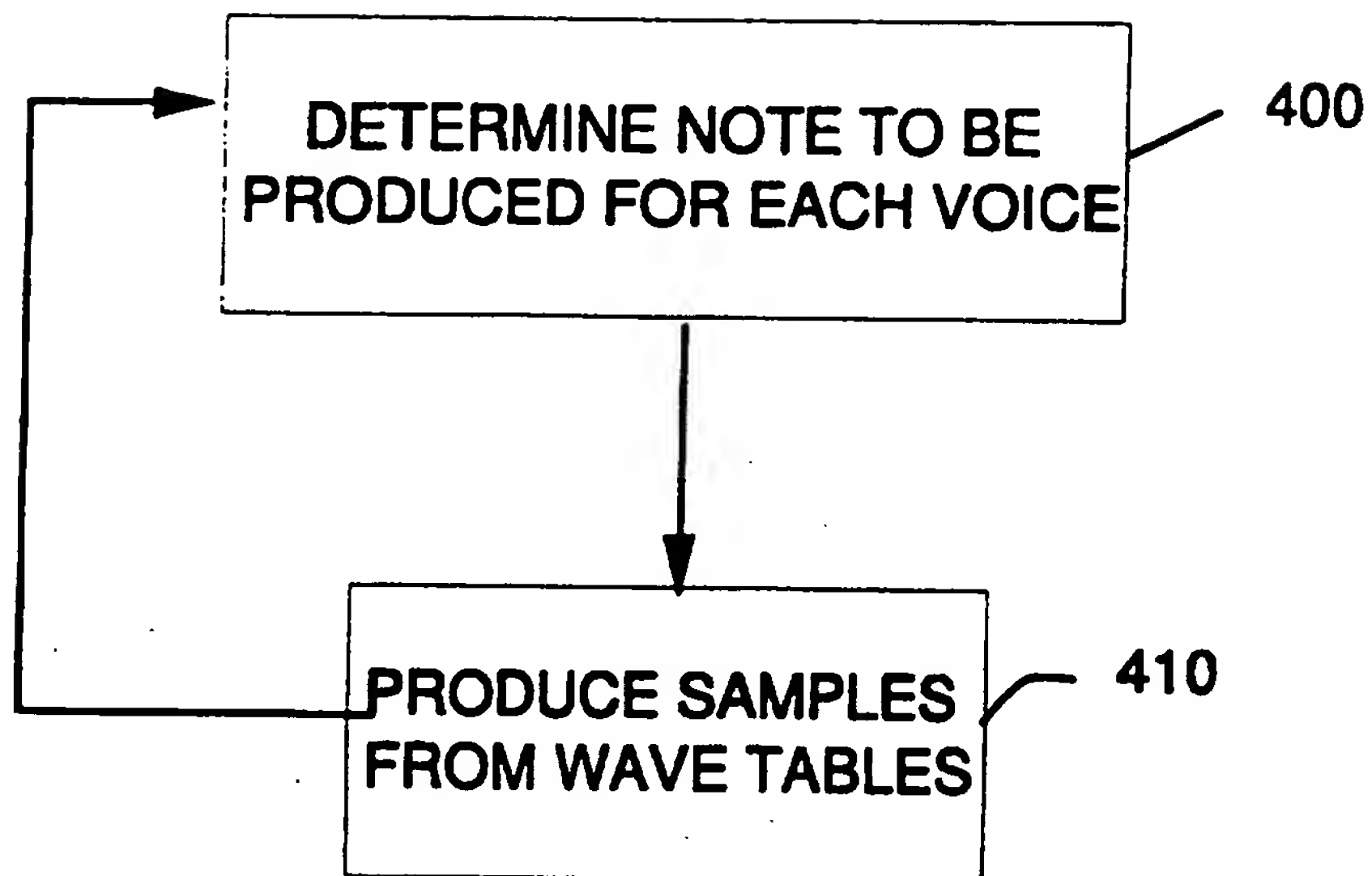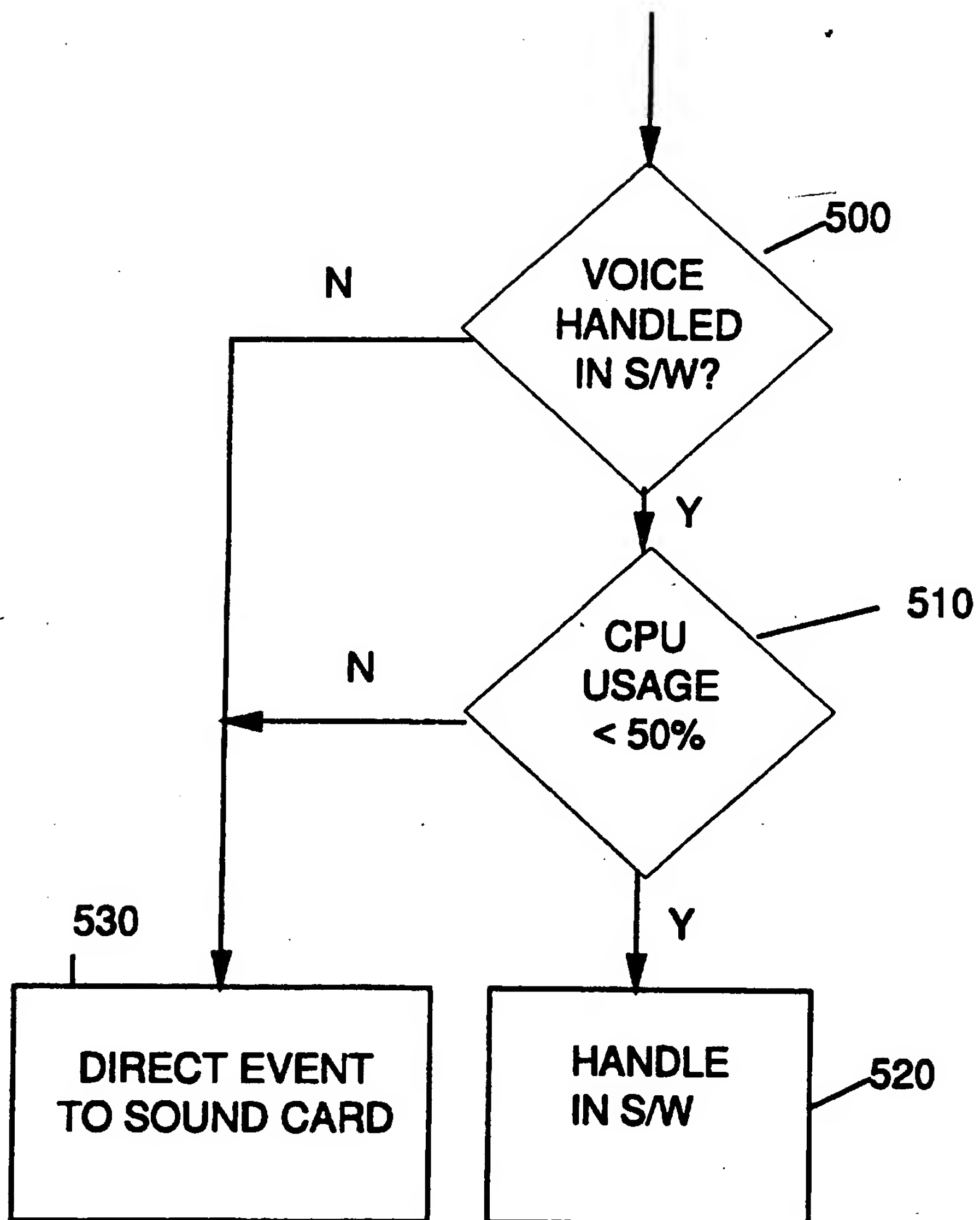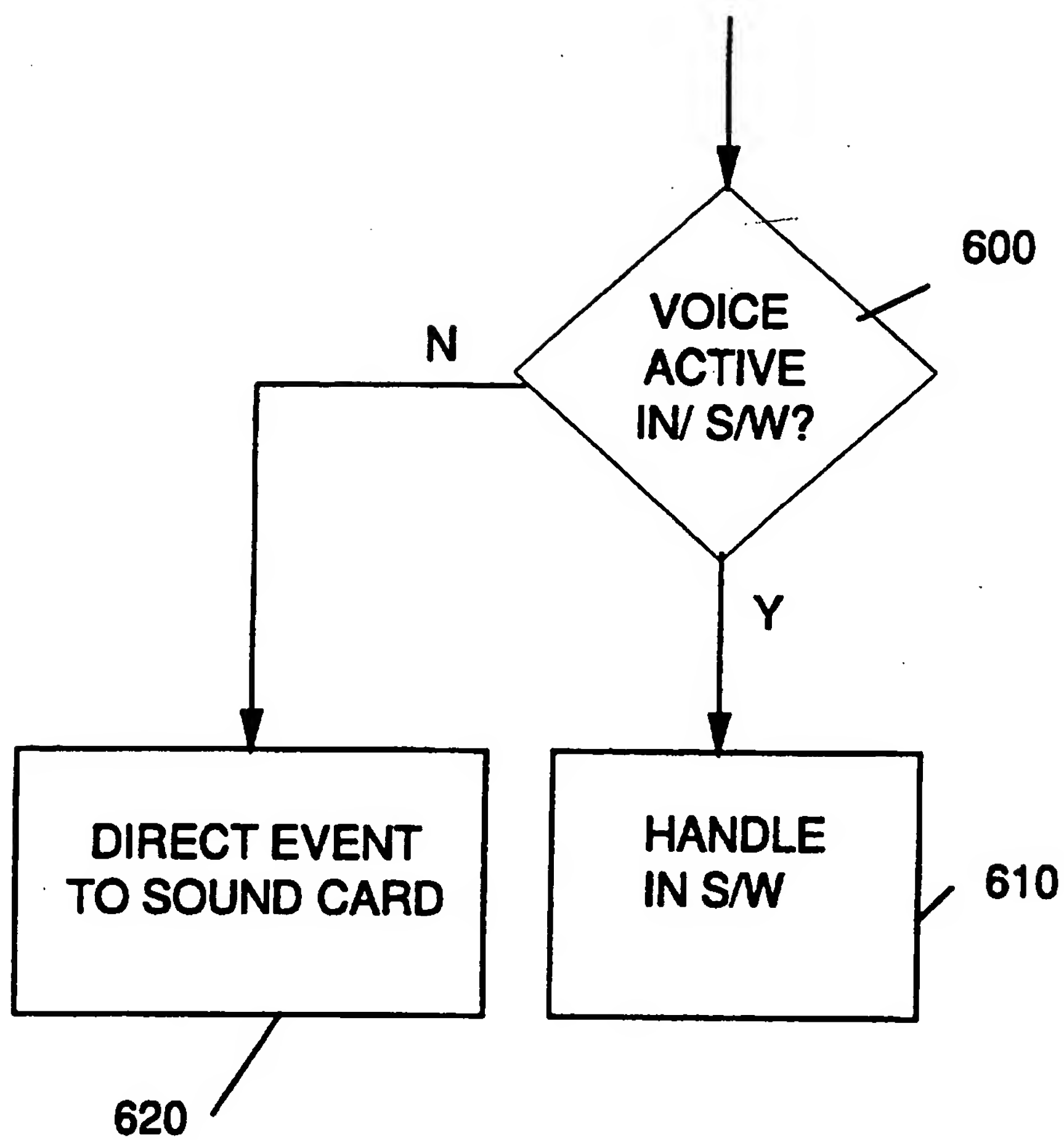 applications. Typically, a dedicated sound card is plugged in the motherboard of the computer, although some recent personal
15   computers have dedicated sound hardware located on the motherboard itself.

The main advantage of such hardware implementations is that even when the CPU of the computer is loaded to the extent of its power, the
20   sound quality is not degraded.

However, an implementation entirely in hardware also has certain disadvantages. For example, the resources of the sound card are only used for the sound generation process and the machine cannot benefit from
25   them in any other way. Software improvements on the computer, for example to the operating system, do not improve the sound quality or the performance since the sound card behaves as a semi-independent entity.

MIDI (Musical Instrument Digital Interface) is an internationally
30   recognised specification for data communication between digital electronic musical instruments and other devices, such as computers, lighting controllers, mixers or the like. The MIDI data specifies performance information, as opposed to sound information. For example, which note or notes are being held down, if any additional pressure is
35   applied to the note after being struck, when the key is released and any other adjustments made to the settings of the instrument. MIDI data is communicated as a serial data stream organised into MIDI 'messages', which contain one MIDI command or event.

40   In a conventional MIDI playback system, a MIDI synthesizer is controlled by a stream of MIDI messages. The synthesizer receives and

decodes the messages and operates accordingly. For example, a 'NOTE ON' event will cause the synthesizer to generate audio samples that correspond to a requested note and velocity that are supplied as parameters. Similarly, a 'NOTE OFF' event will cause the synthesizer to cease generating the audio samples.

Most commercially available sound cards have the capability of acting as MIDI synthesizers by receiving data from MIDI sources either though a MIDI port or via the PC bus.

Software-only MIDI synthesizers have been proposed for use with general purpose computers. These can take advantage of all the resources of the computer, such as CPU power, memory, magnetic and CD-ROM storage, caching mechanisms and virtual memory and are easily customised, upgraded and maintained. However, they have the disadvantage that the sound quality can be degraded when the computing resources they require are not available due to other tasks the computer may be performing. Nevertheless, in recent years the power available from the processors used in personal computers has increased significantly and, in normal use, the load on the CPU can be quite low for much of the time.

DISCLOSURE OF THE INVENTION

This invention is directed to providing an audio synthesizer of the above defined type which combines the advantages of hardware and software implementations.

To achieve this aim, there is now provided an audio synthesizer for generating an analogue or digital audio output in response to coded control instructions representing musical events, the synthesizer comprising: a general purpose computer portion having a CPU programmed to receive the control instructions and generate audio samples; a special purpose hardware portion for receiving the control instructions and generating the audio samples; a controller for directing the control instructions either so that the general purpose computer portion generates the audio samples or so that the hardware portion generates the audio samples; and means to combine the audio samples generated by the general purpose computer portion and the hardware portion to form an audio output which accords with the control instructions.

This provides a hybrid audio synthesizer which combines advantages from both software and hardware implementations and enables the spare computing power of the host CPU to be used to supplement that of the sound card.

5

An embodiment of such a synthesizer has been developed for use with a MIDI data stream, however application of analogous techniques to other forms of control instructions is not excluded.

10   In a preferred embodiment the audio synthesizer comprises measurement means for repeatedly measuring the load on the CPU and the controller is arranged to direct the control instructions according to the measured load on the CPU. The CPU usage can be measured, for example, by timing the synthesis internal loop, which is the computation

15   intensive loop, although other methods for determining the CPU usage are possible.

In this case, when the CPU is not highly loaded, the hybrid synthesizer can handle some of the MIDI events in software using the

20   native CPU, while propagating some of the events to the hardware portion. The decision as to how many simultaneous notes should be handled in software and how many in hardware is based on the current CPU usage.

The general purpose computer can be programmed to act as the

25   controller and to either generate the audio samples itself or to transfer the instructions to the hardware portion. Alternatively, the controller could be implemented as part of the special purpose hardware.

Advantageously, the controller is arranged to direct the

30   intructions according to voice-type. In this way, the audio synthesizer can be arranged so that a predefined set of voices are handled by the general purpose computer portion and thus the quality of the sound can be improved by directing intructions relating to particular instruments either to the software or the hardware synthesizers according to whether

35   they are best suited to the particular synthesis method used.

The audio synthesizer can include delay means for delaying the direction of control instructions to the hardware portion, so that the audio samples generated by the hardware portion are synchronised with the

40   audio samples generated by the general purpose computer.

In one particularly preferred embodiment, the synthesis on the main
CPU can be done using a wave-table synthesis, whilst using a relatively
cheap sound card which only supports FM synthesis. In this case, both
quality of the produced music and the overall performance is greatly
5   improved.

Preferably, the synthesizer includes an input for receiving audio
samples and means for mixing audio samples received at said input with
audio samples generated in response to the coded control instructions.
10   This allows wave messages to be received from other sources and mixed
with the wave data generates by the synthesizer.

In one implementation, the audio synthesizer can take the form of a
personal computer with an add-on sound card, although, of course, many
15   other configurations are possible.

Viewing the invention from a second aspect, there is provided a
method of operating an audio synthesizer to generate an analogue or
digital audio output in response to coded control instructions
20   representing musical events, the synthesizer comprising: a general
purpose computer portion having a CPU programmed to receive the control
instructions and generate audio samples; and a special purpose hardware
portion for receiving the control instructions and generating the audio
samples, the method comprising directing the control instructions either
25   so that the general purpose computer portion generates the audio samples
or so that the hardware portion generates the audio samples so that the
audio samples generated by the general purpose computer portion and the
hardware portion can be combined to form an audio output which accords
with the control instructions.
30

Suitably, it can be arranged that the audio samples can be combined
to accord with the control instructions by delaying the direction of
control instructions to the hardware portion, so that the audio samples
generated by the hardware portion are synchronised with the audio samples
35   generated by the general purpose computer.

In a preferred embodiment the method comprises receiving a NOTE ON
instruction; determining whether the NOTE ON instruction is for a
predefined set of voices to be handled by the general purpose computer
40   portion and, if not, directing the NOTE ON instruction so that the
hardware portion generates corresponding audio samples; measuring the CPU

load and if the CPU load exceeds a predefined threshold, directing the NOTE ON instruction so that the hardware portion generates corresponding audio samples; otherwise directing the NOTE ON instruction so that the general purpose computer portion generates the audio samples.

In this embodiment the method further comprises: receiving a NOTE OFF instruction; determining whether the NOTE OFF instruction relates to a note being handled by the general purpose computer portion and, if not, directing the NOTE OFF instruction to the hardware portion; otherwise, directing the NOTE OFF instruction to the general purpose computer portion.

The invention can be implemented in the form of an article of manufacture comprising a computer usable medium in which program code is embodied for causing a computer to perform the above described methods.

BRIEF DESCRIPTION OF DRAWINGS

An embodiment of the invention will now be described by way of example only with reference to the accompanying drawings, wherein:

Fig 1 is a schematic diagram showing a personal computer;

Fig 2 is a simplified schematic functional block diagram of a sound card;

Fig 3 is a schematic diagram showing the software structure of an audio synthesizer;

Fig 4 is a flow diagram showing the synthesis loop of a software synthesizer;

Fig 5 is a flow diagram showing the process on receipt of a NOTE ON MIDI message;

Fig 6 is a flow diagram showing the process on receipt of a NOTE OFF MIDI message.

BEST MODE(S) FOR CARRYING OUT THE INVENTION

Fig 1 is a schematic diagram showing a personal computer arranged to function as an audio synthesizer. The computer comprises conventional components such as display device 100 and associated display adapter 110, and I/O interface 120 to which is attached a keyboard and a mouse. The computer also comprises a CPU 130, RAM 140 and a magnetic storage device 150. These components are arranged to intercommunicate via a bus 160 in conventional manner.

The computer also comprises an audio adapter 160 which is capable of implementing a MIDI synthesizer by utilising a digital signal processor. Audio adapter 160 is shown connected one or more loudspeakers indicated at 170.

5

The system shown in Fig 1 may be implemented by using an IBM PS/2 computer available from IBM Corporation and a SoundBlaster 16 Value Edition card available from Creative Labs Inc (IBM and PS/2 are trademarks of IBM Corporation and SoundBlaster is a trade mark of 10   Creative Technology Inc).

Fig 2 shows a simplified functional block diagram of the relevant parts of the adapter. It comprises bus interface logic 200, FM synthesizer 210, digital to analog converter 230 and audio amplifiers and 15   mixers 240. Of course, in other embodiments synthesizer 210 could be any other kind of synthesizer, eg a wavetable or waveguide synthesizer. The card has an audio output indicated at 250, though the audio samples might equally be output in digital form for digital recording or processing via an external D/A converter. The structure and general operation of such a 20   card will be well known to those skilled in the art and will not be discussed further herein. The card is capable of simultaneously accepting and combining both MIDI events and wave samples sent from the computer.

25   MIDI data is communicated as a serial data stream organised into MIDI 'messages', which contain one MIDI command or event. MIDI commands are usually composed of one, two, or three bytes of data arranged and transmitted one after another. The first byte sent in each command is called the 'status' byte and specifies an operation to be performed. The 30   next one or two bytes, if used, represent parameters of this command. For example, a NOTE ON command comprises three bytes, the first of which is the status byte. This byte tells a synthesizer to play a note and specifies the channel number. The channel number usually represents the type of sound to be played, ie which instrument of the synthesizer is to 35   be used. The second byte specifies the note to be played and the third byte specifies the velocity value for the note. The bytes of the MIDI commands are specified in the MIDI standard.

In a conventional MIDI playback system a MIDI synthesizer can be 40   controlled by a MIDI sequencer in the following way. A standard MIDI file (SMF) contains a set of events, which are intended to be executed by

a synthesizer at particular times. Generally, the events are not
uniformly spaced in time. A conventional MIDI sequencer parses the
standard MIDI file, reads the present MIDI event and the time difference
between it and the next event. The sequencer then sends the event in a
MIDI message to a MIDI synthesizer at the time it is to be executed. The
sequencer usually sets a timer and reads the next MIDI event after this
time difference has elapsed.

A conventional MIDI synthesizer receives the MIDI message that the
sequencer sends, decodes the message and operates accordingly. For
example, a 'NOTE ON' event will cause the synthesizer to generate audio
samples that correspond to a requested note and velocity that are
supplied as parameters. Similarly, a 'NOTE OFF' event will cause the
synthesizer to cease generating the audio samples.

In this system a software MIDI synthesizer is implemented as
Dynamic Link Library (DLL). This DLL handles some of the notes in
software, while propagating the rest of the notes to the sound card
according to performance and quality criteria.

A schematic diagram of the software components in the present
implementation is shown in Fig 3. The system comprises a MIDI sequencer
application layer 300, an operating system layer 310, a DLL layer
indicated at 320 and a sound card layer 330.

DLL layer 320 comprises 3 main blocks, controller 335, high level
block 340 and low level block 350. Sound card layer 330 comprises a
sound card device driver 360 and the hardware resources of the sound card
which are indicated at 370. Also shown in Fig 3 is an output buffer 380.

The DLL layer 320 is triggered by MIDI messages which are sent by
MIDI sequencer. These messages can be sent directly by application 300
or via the operating system services.

Controller 335 is responsible for the MIDI logic management. In
other words, this block decides which notes are to be propagated to the
sound card and which should be handled entirely by means of software
synthesis. NOTE ON and NOTE OFF messages for notes to be handled on the
sound card are sent, after a suitable delay to maintain synchronisation,
directly to the sound card via line 345. Other messages are sent to both

the sound card and the software synthesizer as will be described in more
detail below.

The decision regarding which notes to handle by software, and which
notes to propagate to the sound card takes the following considerations
into account.

1.   In the special case where the software DLL supports wave-table
synthesis and the sound card supports only FM synthesis, there is an
advantage in handling instruments which cannot be realistically
synthesised by FM synthesis, such as acoustic instruments like violin,
acoustic piano, clarinet etc, in software, whilst propagating to the
sound card notes related to instruments that are not damaged by the FM
synthesis, such as electric piano, electronic synthesizer effects, etc.

2.   If there is insufficient CPU power to handle any further notes in
software, then any further NOTE ON messages received should be propagated
to the sound card. On the other hand if the CPU is relatively idle, it
can participate in the synthesis process and thus improve the overall
performance. In case of overload, the controller 335 is arranged to
dynamically reduce the number of soft-voices, ie the number of notes that
are currently synthesized by means of software, down to a limit of zero
by directing any further NOTE ON messages to the sound card.

3.   The number of empty slots on the sound card. Each empty slot is
capable of playing one note. The number of slots supported by the sound
card represents the maximum number of notes that the sound card can play
simultaneously.

High level block 340 receives the messages which are to be handled
in software and updates the various internal tables which are used by the
synthesizer. It handles conventional functions of a MIDI synthesizer
such as managing voice allocation etc which are not directly relevant to
the present invention and will be well understood by those skilled in the
art.

Low level block 350 is triggered periodically and contains the
synthesis engine which requires intensive computation. This block is
responsible for the synthesis of notes that are not propagated to the
sound card by controller 335. The general operation of the low level
block is shown in Fig 4.   The loop is generally triggered with a time

period of between 1mS and 5mS. Each time the loop is triggered it checks in step 400 for each voice which kind of note is to be produced and looks in the Wave Table database to find the appropriate waveform. The waveform for each active voice is then transformed if necessary to the correct pitch and the waveforms combined to generate output samples - step 410 - at a suitable sampling rate, for example 44100 samples per second, and places them in buffer 380 for retrieval by the device driver 360 in sound card layer 330.

Since the synthesis algorithm is computationally very intensive, it can be used for monitoring the CPU usage. This measure of the CPU usage is available to controller 335 to enable it to decide whether notes should be propagated to the sound card.

There are a number of ways that the CPU usage can be monitored via the synthesis process. For example, a time measurement can be taken each time the low-level block 340 is triggered. If the difference between the current time and the time that was measured on the previous iteration is consistently larger than the requested period, then it can be assumed that the CPU is overloaded.

Alternatively, a time measurement can be taken at the beginning and it the end of the synthesis process in each iteration. The difference between these two measurements is approximated to be the CPU time which is used for the current number of soft voices. The CPU usage in percentage is then calculated as the ratio of the measured synthesis time and the time interval which is used as the triggering period for the low level block 350.

Controller 335 directs all MIDI messages, apart from NOTE ON and NOTE OFF messages to both the sound card and the high level block 340. In the case of NOTE ON and NOTE OFF messages, the processes shown in Figs 5 and 6 respectively are performed.

In the case of a NOTE ON command it is determined in step 500 whether the instrument to which the note relates is one which a priori should be handled in software because of its nature. If so, then a determination of the CPU load is made in step 510 and if the CPU load is less than a fixed limit the NOTE ON message is directed to high level block 340 and the note is handled in software - step 520. If the CPU is greater than 50% loaded or the instrument to which the note relates can

be adequately handled by the sound card, then the NOTE ON command is directed to the sound card - step 530.

On receipt of a NOTE OFF command, it is determined, in step 600 in Fig 6, whether the voice concerned is presently being handled in software. This information is available in the tables maintained by high level block 340. If the note is being handled in software, then the NOTE OFF message is directed to high level block 340 - step 610, if not then the NOTE OFF message is directed to the sound card - step 620.

It will be appreciated that there is a need to delay propagation of MIDI events to be handled in the sound card to maintain synchronisation between the samples produced in software and those produced in hardware. This will be a constant, but processor and/or sound card-dependant delay and in the present arrangement can be set manually by the user via a suitable menu presented to them. Determination of the correct delay may be easily established empirically by the user.

The present implementation has been written to work under the well known Windows operating system from Microsoft Corp (Windows is a trademark of Microsoft Corp.). The general operation of the system is as follows. First the user invokes a Windows application, such as a sequencer, which requires a MIDI synthesizer. The application opens the MIDI synthesizer DLL via the MIDI Mapper program, with the intervention of the MMSYSTEM driver.

The software MIDI Synthesizer DLL opens the sound card twice - once as a MIDI device to which MIDI messages will be propagated and once as a WAVE device to which the wave samples produced by the software MIDI synthesizer will be sent.

Since the software synthesizer locks the WAVE input of the sound card, it is itself provided with a WAVE input to receive WAVE messages from other sources and handle the mixing of these with the wave data it generates. MIDI messages for instruments for which the sound quality is acceptable using FM synthesis are propagated to the sound card and handled in hardware.

The software MIDI synthesizer handles the MIDI messages for instruments which require Wave-table synthesis up to a fixed limit, such as 50%, of the available CPU power. In case of overload, the software

synthesizer dynamically reduces the number of soft-voices, ie the number
of notes that are currently synthesized by means of software down to a
limit of zero. As described above, this is achieved by directing any
further NOTE ON messages to the sound card.

5

        As will be clear from the above description, the present
implementation takes the form of a computer program and can be
distributed in the form of an article of manufacture comprising a
computer usable medium in which suitable program code is embodied for
10   causing a computer to perform the function of controller 335 described
above. The program may include the high level block 340 and 350 and low
level blocks of the software synthesizer or could be implemented for use
with a preexisting software synthesizer.

15   However, it will be appreciated that many variations are possible
within the scope of the attached claims. For example, the controller
might be implemented on the sound card itself, either in hardware or
software, and arranged to receive MIDI messages from an external source,
the sound card handling some notes itself and directing some notes over
20   the PC bus to a software synthesizer.

INDUSTRIAL APPLICABILITY

        The invention is applicable in the technical field of computers and
25   digital audio systems.

CLAIMS

1.   An audio synthesizer for generating an analogue or digital audio
output in response to coded control instructions representing musical
events, the synthesizer comprising:

     a general purpose computer portion having a CPU programmed to
receive the control instructions and generate audio samples;

     a special purpose hardware portion for receiving the control
instructions and generating the audio samples;

     a controller for directing the control instructions either so that
the general purpose computer portion generates the audio samples or so
that the hardware portion generates the audio samples; and

     means to combine the audio samples generated by the general purpose
computer portion and the hardware portion to form an audio output which
accords with the control instructions.

2.   An audio synthesizer as claimed in claim 1 comprising measurement
means for repeatedly measuring the load on the CPU and wherein the
controller is arranged to direct the control instructions according to
the measured load on the CPU.

3.   An audio synthesizer as claimed in claim 1 wherein the measurement
means comprises timing means for timing a synthesis loop in the program
for generating the audio samples.

4.   An audio synthesizer as claimed in any preceding claim wherein the
general purpose computer is programmed to act as the controller and to
either generate the audio samples itself or to transfer the instructions
to the hardware portion.

5.   An audio synthesizer as claimed in any preceding claim in which the
controller is arranged to direct the instructions according to voice
type.

6.   An audio synthesizer as claimed in any preceding claim in which the
general purpose computer comprises a wave-table synthesizer for

generating the audio samples and the hardware portion comprises an FM synthesizer for generating the audio samples.

7.   An audio synthesizer as claimed in any preceding claim including delay means for delaying the direction of control instructions to the hardware portion, so that the audio samples generated by the hardware portion are synchronised with the audio samples generated by the general purpose computer.

8.   An audio synthesizer as claimed in any preceding claim including an input for receiving audio samples and means for mixing audio samples received at said input with audio samples generated in response to the coded control instructions.

9.   An audio synthesizer as claimed in any preceding claim in the form of a personal computer with an add-on sound card.

10.   A method of operating an audio synthesizer to generate an analogue or digital audio output in response to coded control instructions representing musical events, the synthesizer comprising: a general purpose computer portion having a CPU programmed to receive the control instructions and generate audio samples; and a special purpose hardware portion for receiving the control instructions and generating the audio samples, the method comprising directing the control instructions either so that the general purpose computer portion generates the audio samples or so that the hardware portion generates the audio samples so that the audio samples generated by the general purpose computer portion and the hardware portion can be combined to form an audio output which accords with the control instructions.

11.   A method as claimed in claim 10 comprising delaying the direction of control instructions to the hardware portion, so that the audio samples generated by the hardware portion are synchronised with the audio samples generated by the general purpose computer.

12.   A method as claimed in claim 10 or claim 11 comprising
        receiving a NOTE ON instruction;
        determining whether the NOTE ON instruction is for a predefined set of voices to be handled by the general purpose computer portion and, if not, directing the NOTE ON instruction so that the hardware portion generates corresponding audio samples;

measuring the CPU load and if the CPU load exceeds a predefined
threshold, directing the NOTE ON instruction so that the hardware portion
generates corresponding audio samples;

otherwise directing the NOTE ON instruction so that the general
purpose computer portion generates the audio samples.

13.   A method as claimed in any of claims 10 to 12 comprising
receiving a NOTE OFF instruction;
determining whether the NOTE OFF instruction relates to a note
being handled by the general purpose computer portion and, if not,
directing the NOTE OFF instruction to the hardware portion;
otherwise directing the NOTE OFF instruction to the general purpose
computer portion.

14.   An article of manufacture comprising a computer usable medium in
which program code is embodied for causing a computer to perform a method
as claimed in any of claims 10 to 13.

Application No: GB 9520124.0	**Examiner:** R. F. King
Claims searched: 1 to 13	**Date of search:** 12 January 1996

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G5J[JEBA,JECM,JEGA,JEGH,JEGK,JEGX,JESS,JESX,JER]

Int Cl (Ed.6): G10H 1/00, 5/00 - 5/14, 7/00 - 7/12

Other: ONLINE: WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
	None	1, 10